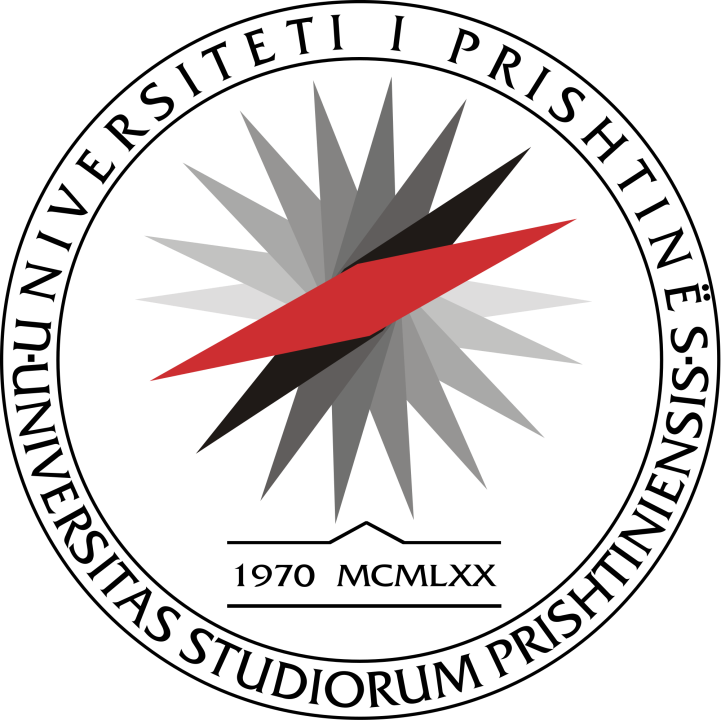
Universiteti i Prishtinës “Hasan Prishtina”

Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike

Departamenti i Kompjuterikës



**RAPORT: DIZAJNIMI KLIENT-SERVER**

**LËNDA: Rrjeta Kompjuterike**

Profesori i lëndës: Prof. Blerim Rexha Studenti: Flutura Haxhaj

Asistenti i lëndës: M.Sc.Haxhi Lajqi Nr.ID-së: 170714100080

Prishtinë, 2019

Përmbajtja

[Abstrakt 3](#_Toc511163350)

[Hyrje 4](#_Toc511163351)

Përshkrimi i librarive dhe metodave të përdorua(Rezultatet e testimit TCP)..............................................5

Rezultatet e testimit UDP-Protokol........................................................................................................... 11

[Kodi 16](#_Toc511163354)

[Source Code per ServerSocketTCP 16](#_Toc511163355)

[Source Code per ClientSocketTCP 20](#_Toc511163356)

[Source Code per ServerSocketUDP 21](#_Toc511163357)

[Source Code per ClientSocketUDP 25](#_Toc511163358)

[Referencat 26](#_Toc511163359)

# Abstrakt

Rrjetat kompjuterike janë të destinuara që të mundësojnë komunikimin me anë të pajisjeve.

Një rrjete kompjuterik paraqet bashkësi të dy apo më tepër kompjuterëve, të cilët janë të lidhur ndërmjet veti për të këmbyer të dhënat dhe për të shfrytëzuar resurset e përbashkëta. Rrjetëzimi është zhvilluar nga nevoja që më shumë shfrytëzues njëkohësisht të shfrytëzojnë të dhënat si nga kompjuteri i tij ashtu edhe nga kompjuterët tjerë në rrjetë.  
Komunikimi në rrjeta kompjuterike bëhet nëpërmjet soketave (ang. sockets), në rastin tonë ne krijojmë socket në gjuhën programuese Python, socketi krijohet në IP Adresën dhe Portin e caktuar. Një socket në mënyrë unike ka një IP-adresë dhe një port. Me krijimin e socketit kemi një kanal komunikues në mes të klientit dhe serverit te cilin mund ta përdorim për dërgimin apo pranimin e informatave në klient apo në server. Projekti nga lënda “Rrjeta kompjuterike” ka pikërisht këtë për qëllim, që të implementohen metodat dhe njohuritë programuese që kemi fituar gjatë ligjëratave dhe ushtrimeve për të krijuar një server dhe një klient.

Në rastin tonë kemi detyrën për një komunikim të rregullt me komanda në mes të klientit dhe serverit ku klienti parashtron një pyetje (metodë) së bashku me parametrat përkatës nëse janë të nevojshëm të cilen e merr serveri dhe në bazë të algoritmit zgjedh problemet e caktuara dhe i përgjigjet klientit. Kjo mund të kryhet në dy protokolle në TCP dhe në UDP.

Dallimi qenësor në mes te TCP-së dhe UDP-së është se TCP fillimisht krijon lidhje server-klient, ndërsa UDP nuk e kërkon këtë, që njëkohësisht rrit shpejtësinë dhe rrit rreziqet që të dhënat të shkëmbehen pa garancion të plotë.

Pas zhvillimit të klientit dhe serverit, në raport rezultatet tregojnë se secila metodë e parashtruar në detyrë mund ti dërgohet serverit dhe ai do të përgjigjet saktësisht.

Ekzekutimi i klientit dhe serverit bëhet duke përdorur cmd (Command Line) ku bëhet edhe komunikimi ndërmjet tyre.

# Hyrje

Komunikimi është prej çështjeve kyçe në rrjeta kompjuterike. Dizajnimi i një serveri dhe klienti është mënyra se si të arrihet ai komunikim.  
Në kuadër të lëndës së Rrjetave Kompjuterike kemi zhvilluar një projekt, i cili krijon dhe dizajnon socket-at në anën e serverit dhe në anën e klientit.

Ky projekt është zhvilluar në gjuhën programuese Python, me anë të së cilit së pari krijohet një socket në pjesën e klientit dhe në pjesën e serverit, socketi krijohet duke i dhënë parametrat përkatës në rastin tonë parametrin e parë për krijimin e socketit e përdorim variablën AF\_INET të klasës socket e cila tregon se për komunikim do të përdoret (familja e ip adresave) IPv4, parametri i dytë cakton se a do të përdoret protokolli TCP apo UDP, për TCP kemi përdorur variablën SOCK\_STREAM kurse për UDP kemi përdorur SOCK\_DGRAM. Socketi krijohet në të dy anët edhe tek serveri i cili pret për lidhje edhe tek klienti i cili provon të krijojë lidhje me serverin e caktuar. Në server definohen të gjitha metodat dhe algoritmet të cilat kryejnë punë të caktuar varësisht nga kërkesa e klientit nëse kërkesa e klientit nuk është valide atëherë serveri do të kthej mesazh duke treguar që komanda nuk ekziston, kurse tek klienti krijohet socketi dhe tregohen metodat të cilat mund të përdoren. Sikur që është treguar më parë projekti është zhvilluar më parë në gjuhën programuese Python , sistemi operativ në të cilin është ekzekutuar është Windows 10 (64-bit), kurse si IDE është përdorur Visual Studio 2019. Në server është definuar metoda client\_handler e cila krijohet secilën herë në thread të ri që do të thotë se secili klient e ka metodën e vet ku komunikon me server, kjo metodë pranon si parametra socketin dhe adresën (ip dhe port) të klientit, të cilat merren nga funksioni accept() i socketit të serverit, serveri në threadin kryesor ekzekutohet në një unazë while në të cilën priten lidhjet e reja nga klientët dhe në momentin kur pranohet një lidhje ruhet socketi dhe adresa dhe krijohet threadi i ri me metodën start\_new\_thread e cila pranon dy parametra, i pari është metoda e cila do të ekzekutohet në threadin e ri dhe parametri i dytë është një listë e parametrave të cilat i pranon metoda e cekur në parametrin e parë. Limitimi i qasjes së klientave në server në të njejtën kohë është bërë përmes metodës listen(nr. i klientave) të metodës socket, metoda listen përdoret vetëm në protokollin TCP pasi që në protokollin UDP nuk kemi qasje të përhershme në mes të klientit dhe serverit.

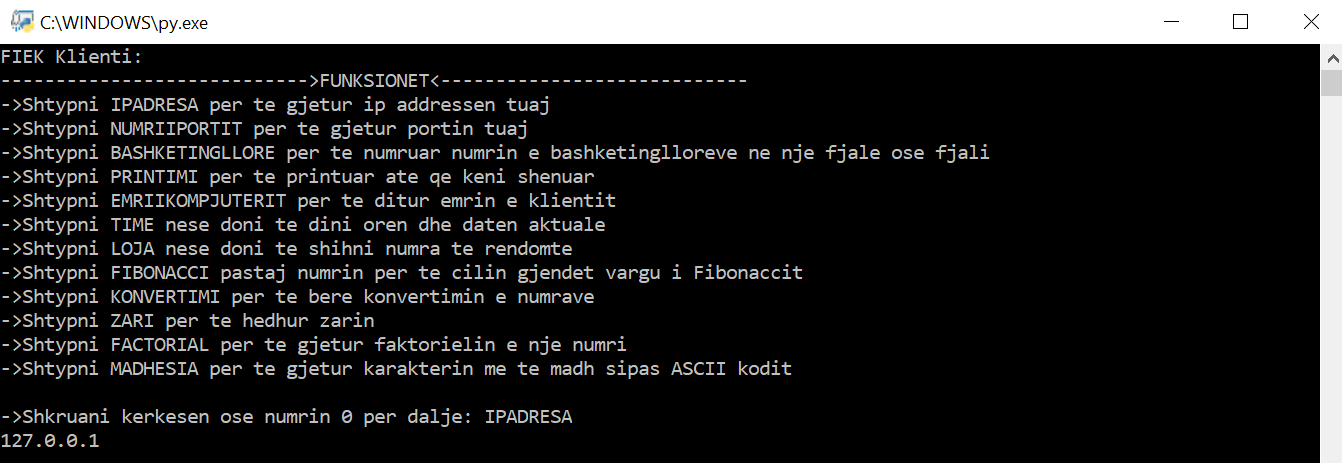
# Përshkrimi i librarive dhe metodave te perdorura

Përveç librarisë socket për projekt gjithashtu janë përdorur këto librari:

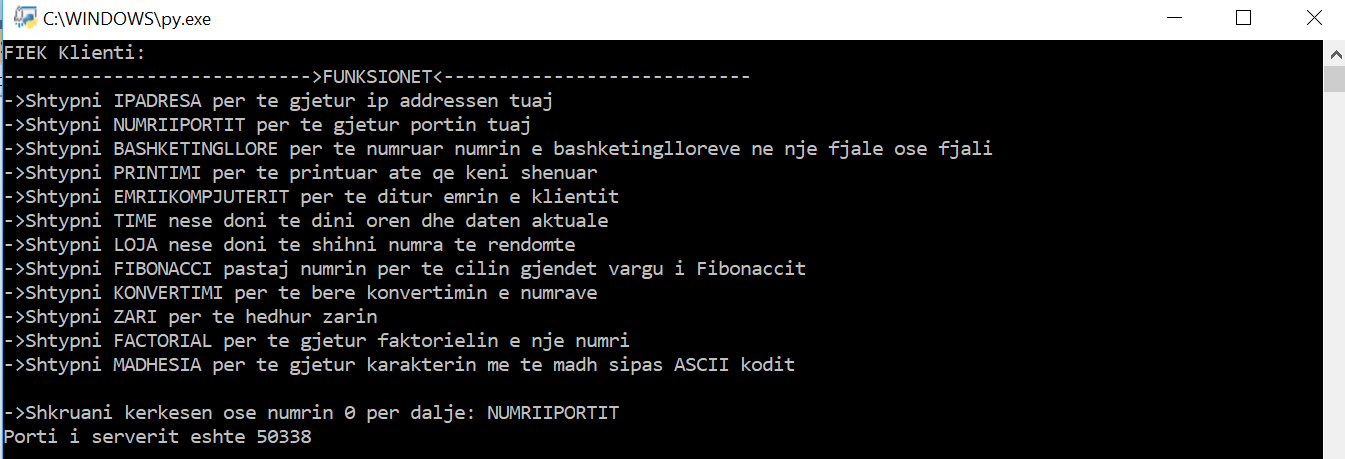
* “\_thread”: për përdorimin e metodës start\_new\_thread
* “datetime”: për gjetjen e kohës në komandën TIME
* “random”: për komandën LOJA dhe për komandën ZARI
* “math”: për importimin e funksioneve matematikore

**Metodat e definuara për komanda dhe testimi ne command line :**

**ipadresa** - kthen ip adresën e klientit. Kjo metodë pranon adresën e klientit me të cilën është lidhur në socket dhe nga ajo merr anëtarin e parë.

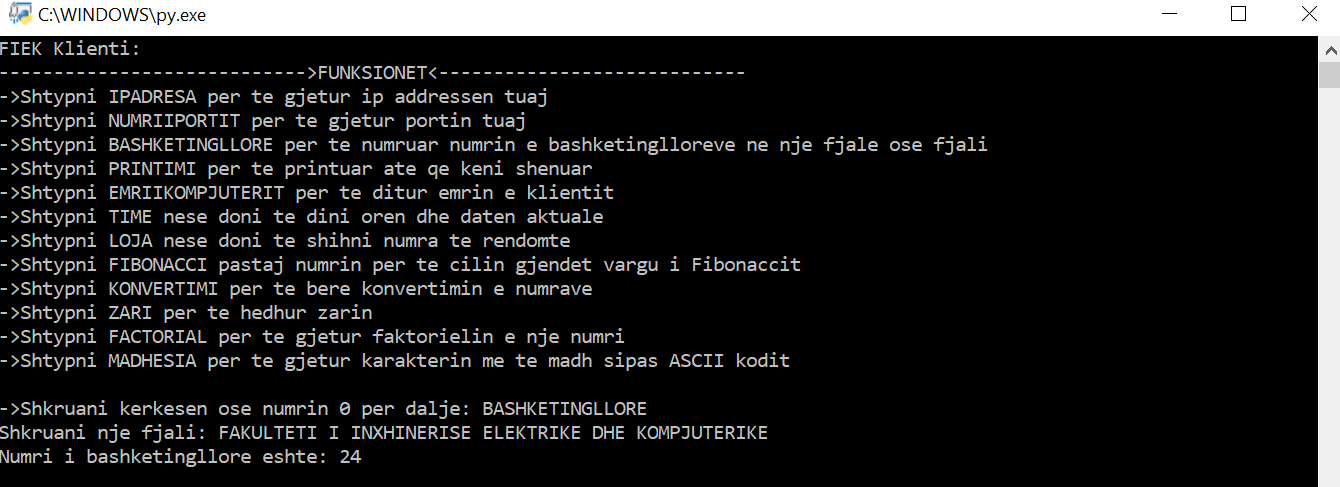


**numriiportit** – kthen portin e klientit. Kjo metodë pranon adresën e klientit e cila është marrur nga metoda accept() kur është pranuar lidhja dhe nga ajo merr anëtarin e dytë i cili është porti.

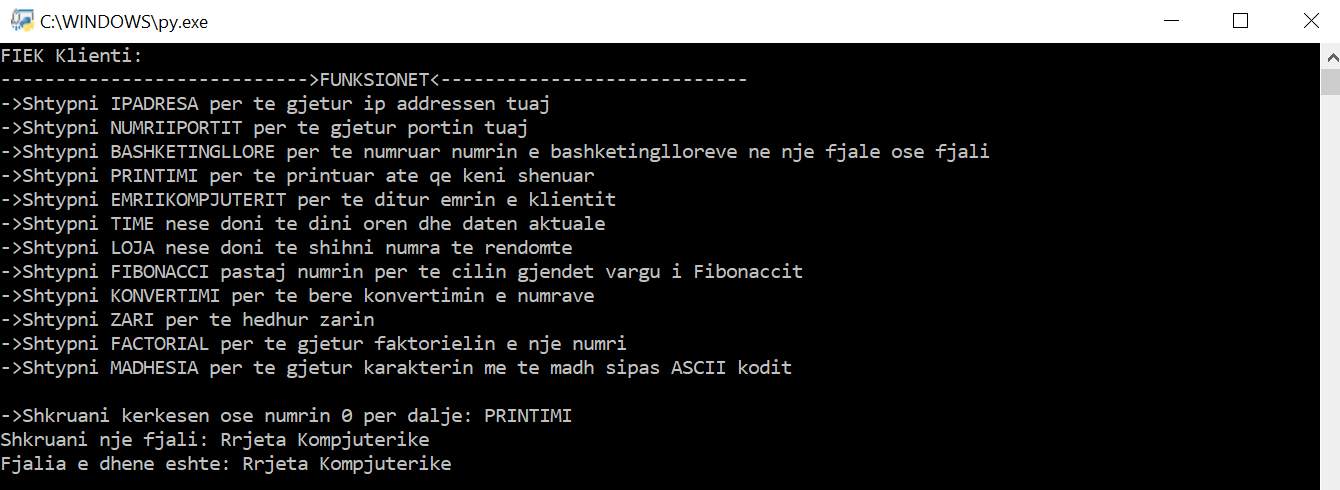


**bashketingllore** – kjo metodë pranon si parameter një string dhe duke përdorur aftësinë e stringut si listë e karaktereve hyn në unazë duke krahasuar secilën shkronjë me një listë të vetë definuar e cila përmban të gjitha bashketinglloret e mundshme dhe sa herë që një krahasim është i saktë një variabël rritet për një duke filluar nga 0 dhe në fund kthen vlerën e asaj variable.

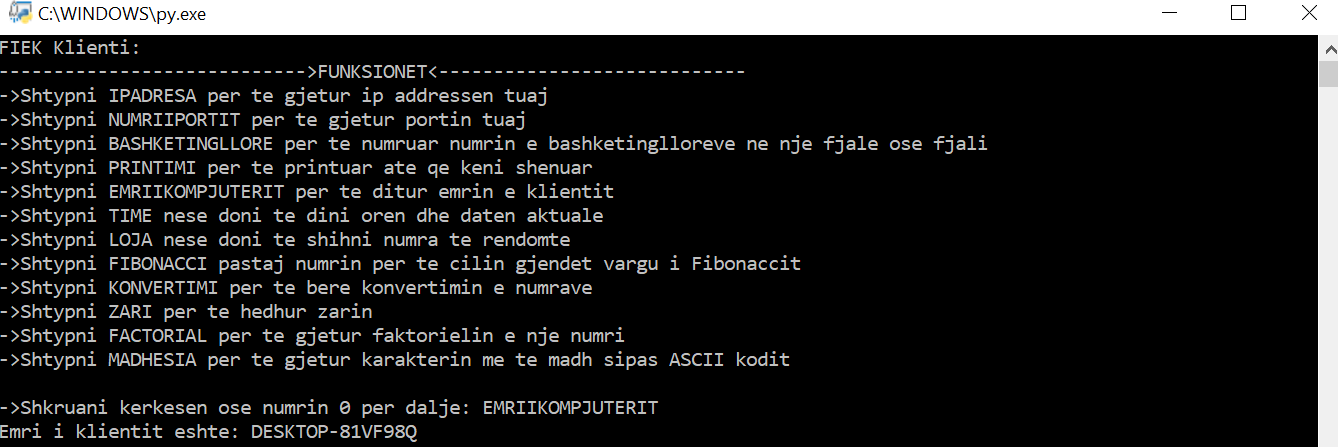
Metoda nuk është Case-Sensitive, pra klienti mundet të shkruajë fjalët edhe me shkronja të vogla edhe me shkronja të mëdha.



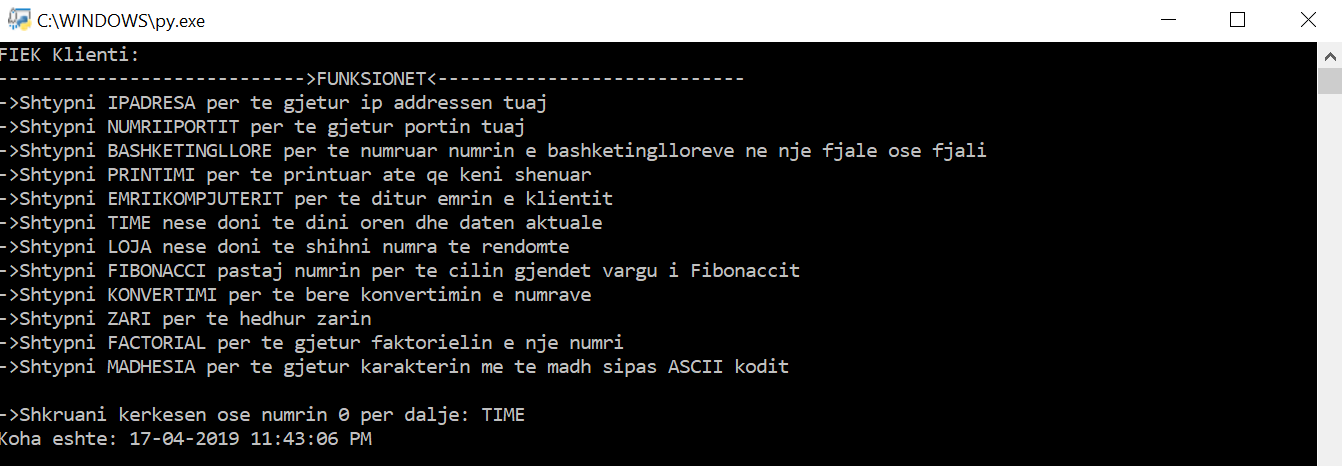
**printimi** – kjo metodë dallon nga TCP dhe UDP në TCP së pari klienti shkruan komandën kurse serveri e pret për një informacion të dytë të cilën e jep klienti dhe atë e kthen mbrapa tek klienti ku klienti e printon atë në Command Line duke përdorur metodën print(var), kurse në protokollin UDP komanda jepet me hapësirë në mes të komandës printo dhe fjalisë e cila do të printohet (p.sh: printo abc abc), së pari duke përdorur metodën split(‘ ‘) të cilës i kemi dhënë si argument një delimiter që në këtë rast është hapësira ndahet në një listë e cila pastaj fillon në një unazë FOR nga elementi i dytë i listës dhe bashkangjet secilin element pas tij në një string me hapësira për nxjerrjen e saktë të fjalisë së dhënë.



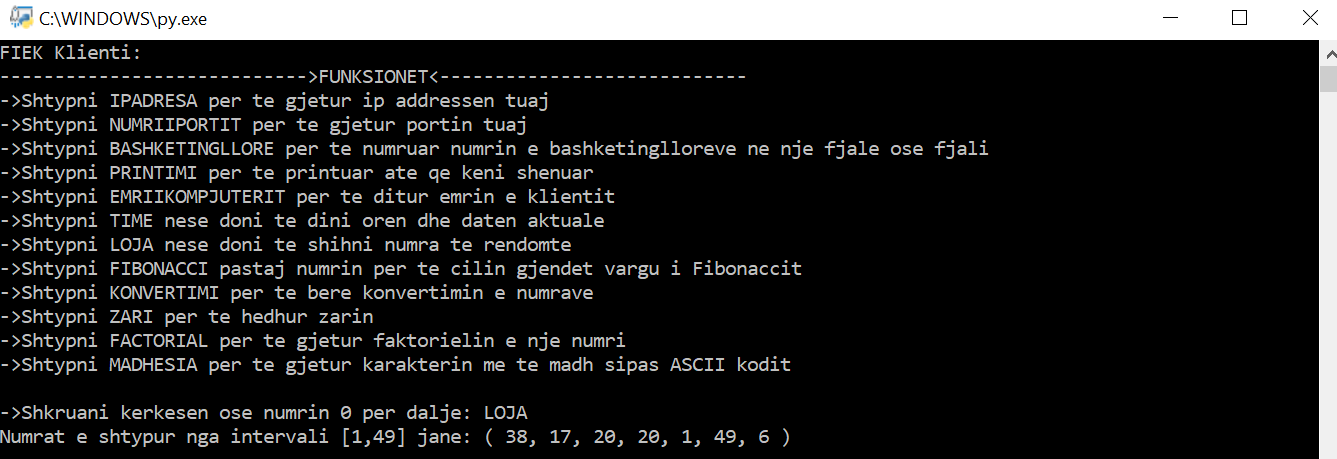
**emriikompjuterit** – kthen emrin e hostit të klientit. Metoda përdorë funksionin e socketave gethostbyaddr dhe merr vetëm anëtarin e parë.

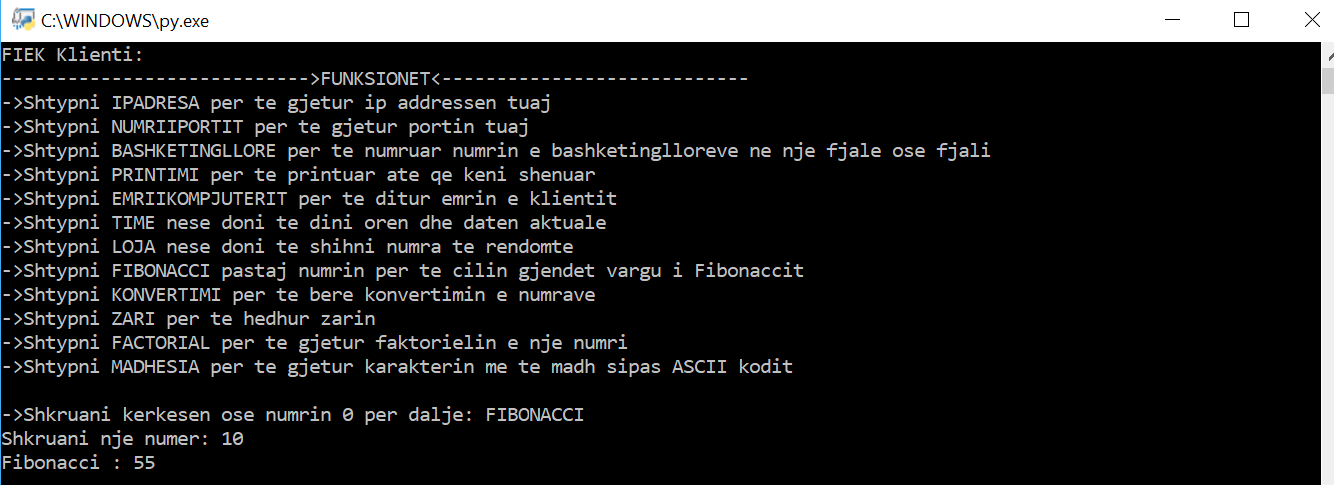


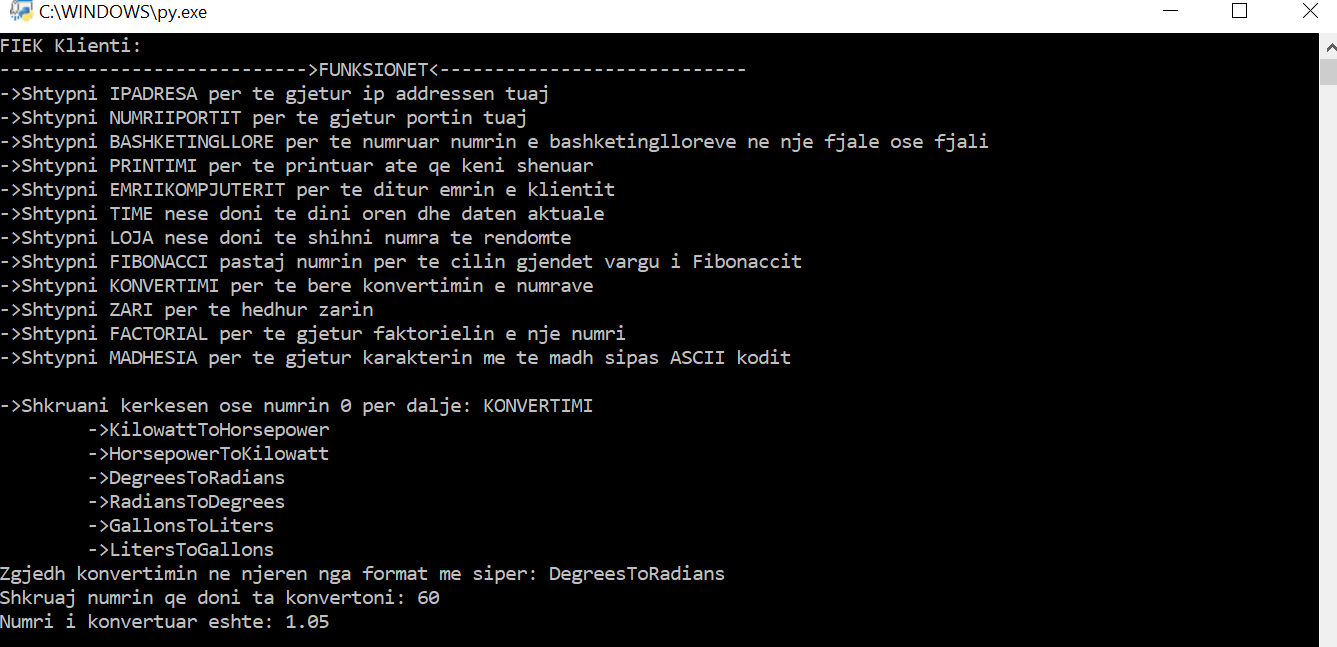
**time** – kthen kohën aktuale të serverit. Kjo metodë përdor librarinë datetime dhe metodën build-in të klasës datetime now().



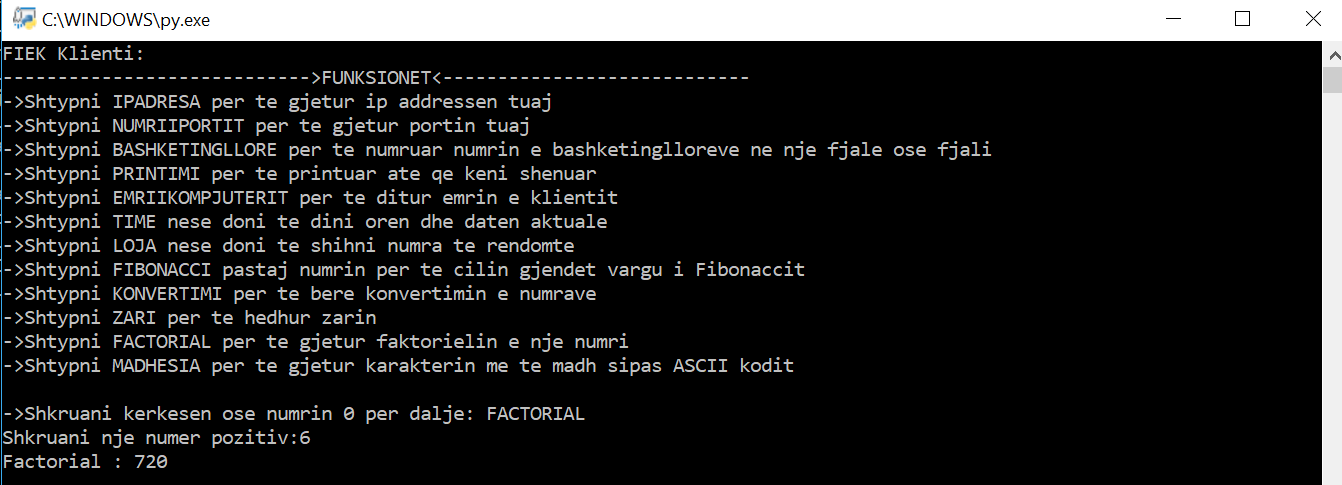
**loja** – kthen 7 numra të rastësishëm duke përdorur klasën random. Kjo metodë përdor librarinë random dhe metodën randint duke ia caktuar rangun prej 1-49. Brenda një unaze e cila ekzekutohet 7 herë, në secilin ekzekutim të saj gjenerohet një numër random i cili ruhet brenda një stringu. Pastaj kthehet kjo listë e numrave e ruajtur si string.



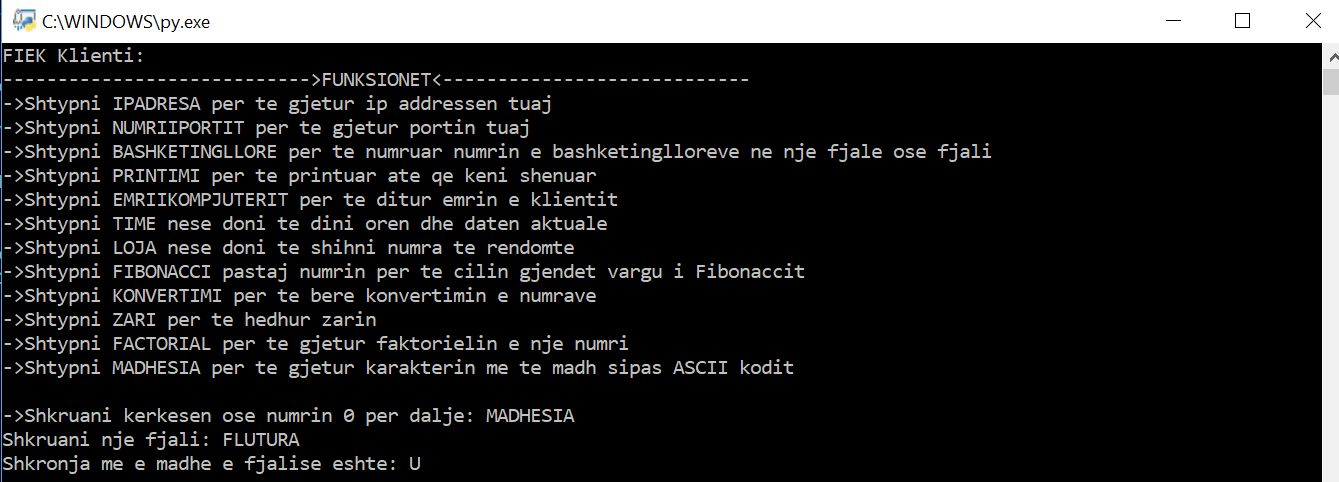
**fibonacci** – kthen numrin fibonacci të numrit përkatës. Metoda fillimisht shqyrton parametrin e pranuar se a është numër pozitiv. Nese plotësohet ky kusht metoda kthen 1 në rastin kur numri për të cilin kërkohet fibonacci numri është 1 apo 2. Kurse në rastet tjera numri fibonacci gjenerohet nga një unazë e cila shkon duke i mbledhur numrat fibonacci derisa të gjeneroj numrin fibonacci për parametrin përkatës.   


**konvertimi** – kjo metodë mund të kryejë konvertime të ndryshme si p.sh. nga Degrees në Radian, Radian në Degrees, Gallons në Liters etj. Metoda pranon një tekst si parametër, i cili definon opsionin që e ka caktuar klienti dhe varësisht nga kërkesa ekzekutohet ndonjëri prej kushtëzimeve të përdorura dhe rezultati ruhet në një variabël duke u llogaritur varësisht prej kërkesës dhe duke përdorur formulat e gatshme për rastet përkatëse.  


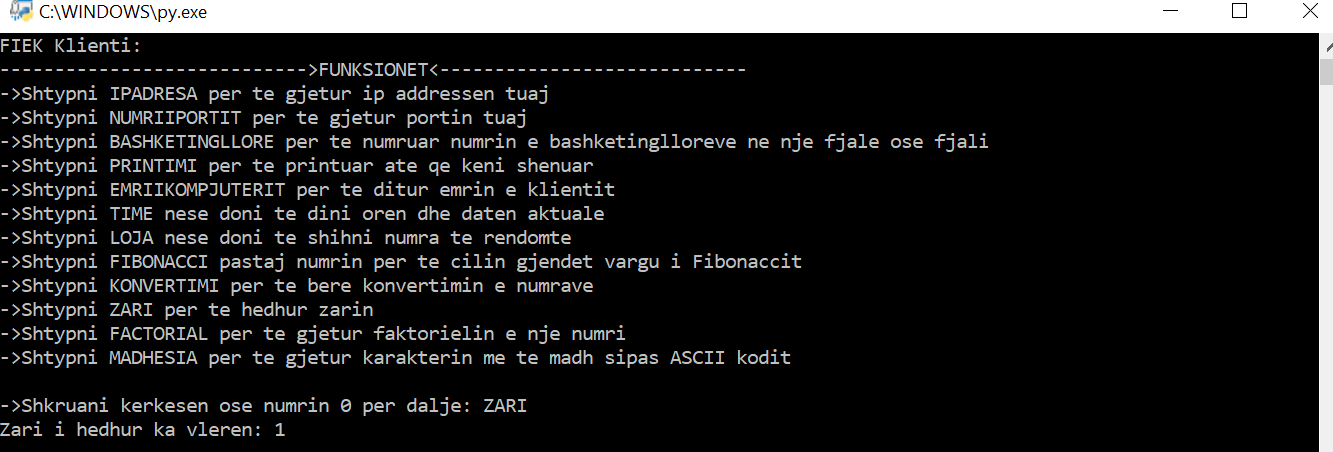
**factorial** – kjo metode llogarite faktorielin e një numri të plotë pozitiv n, i shënuar me n!, faktorieli i numrit natyral *n* është funksion me të cilin paraqitet prodhimi i numrave natyral duke filluar nga 1 deri te numri i caktuar *n*.



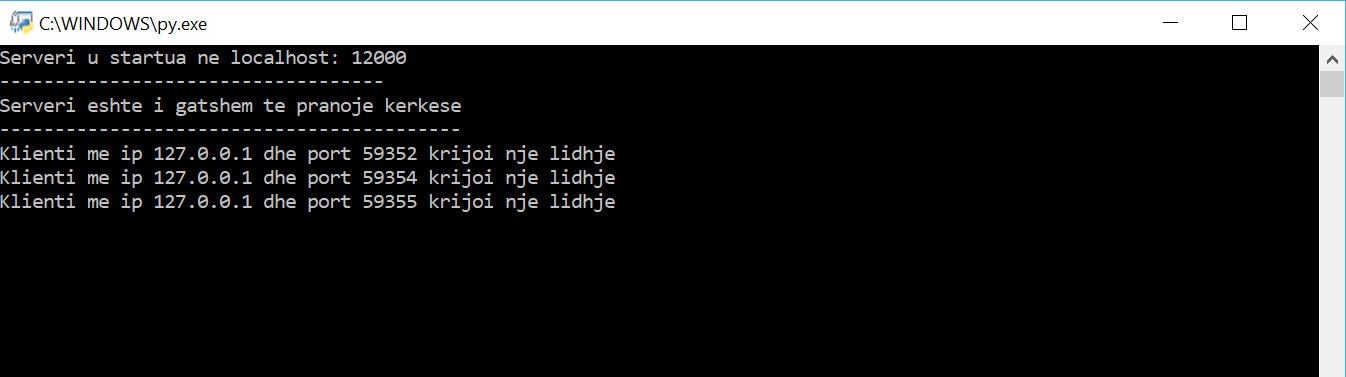
**madhësia** – kjo metodë pranon si parametër një fjali, fjalisë i qasemi me një unazë for duke shikuar secilin karakter a është më i madh se një shkronjë testuese fillestare ‘A’, kushti që do të plotësohet bëhet duke përdorur funksionin ord() e cila kthen vleren e karakterit të dhënë në parametër sipas kodit ASCII dhe e krahason karakterin testues me secilin karakter prej fjals së dhënë, nëse kushti plotësohet karakteri testues mbishkruhet me karakterin e tanishëm dhe në fund kthen shkronjën e cila është më e madhja nga ASCII kodi.



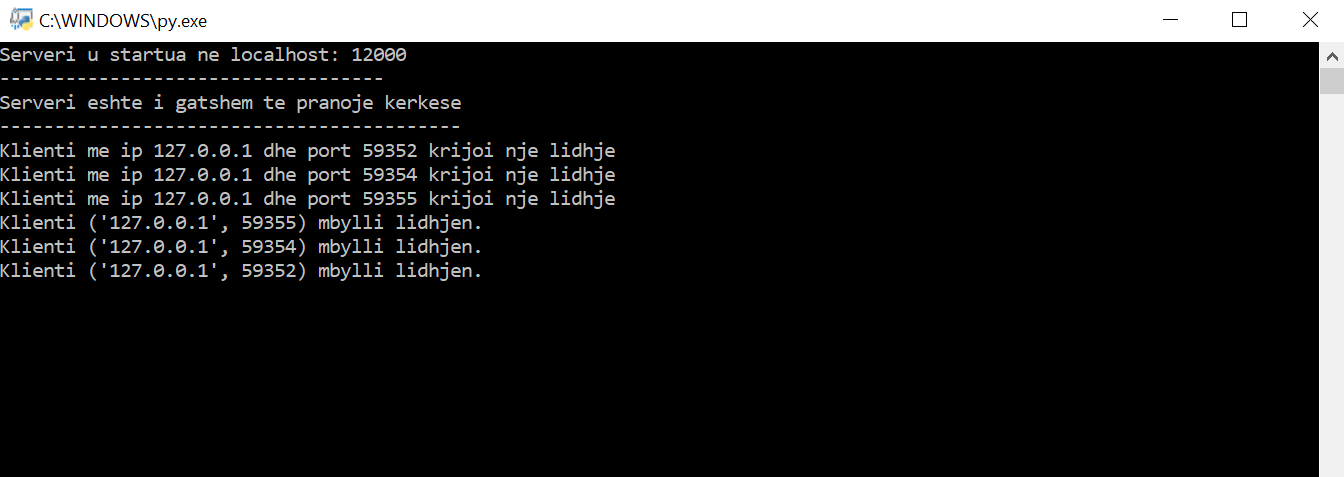
**zari** – kjo metodë kthen një numër të rastësishëm prej numrit 1 deri në 6 duke përdorur metodën randint() të klasës random.



**Krijimi i Lidhjes**

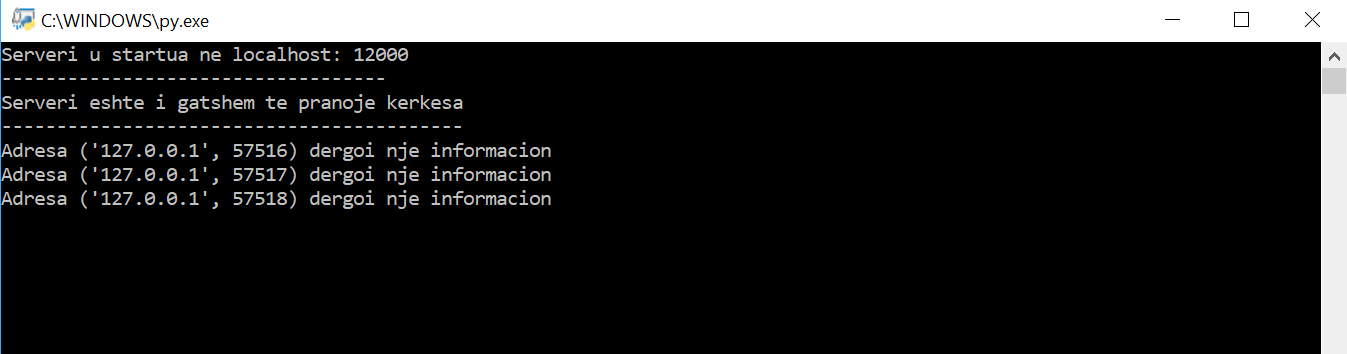


**Mbyllja e Lidhjeve**

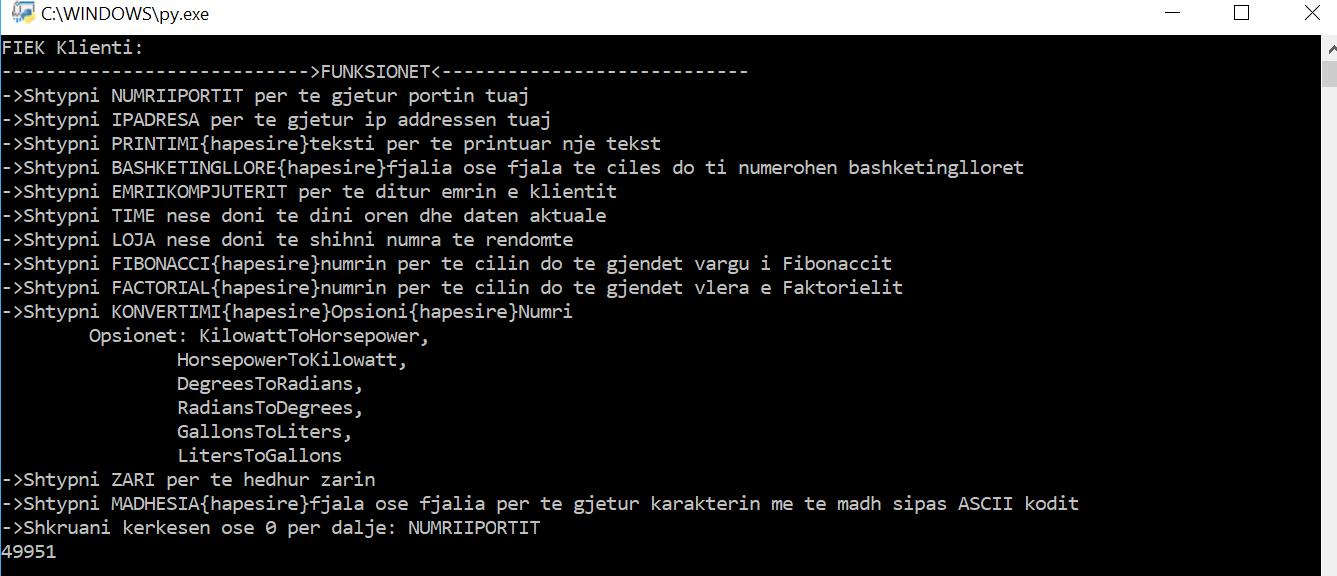


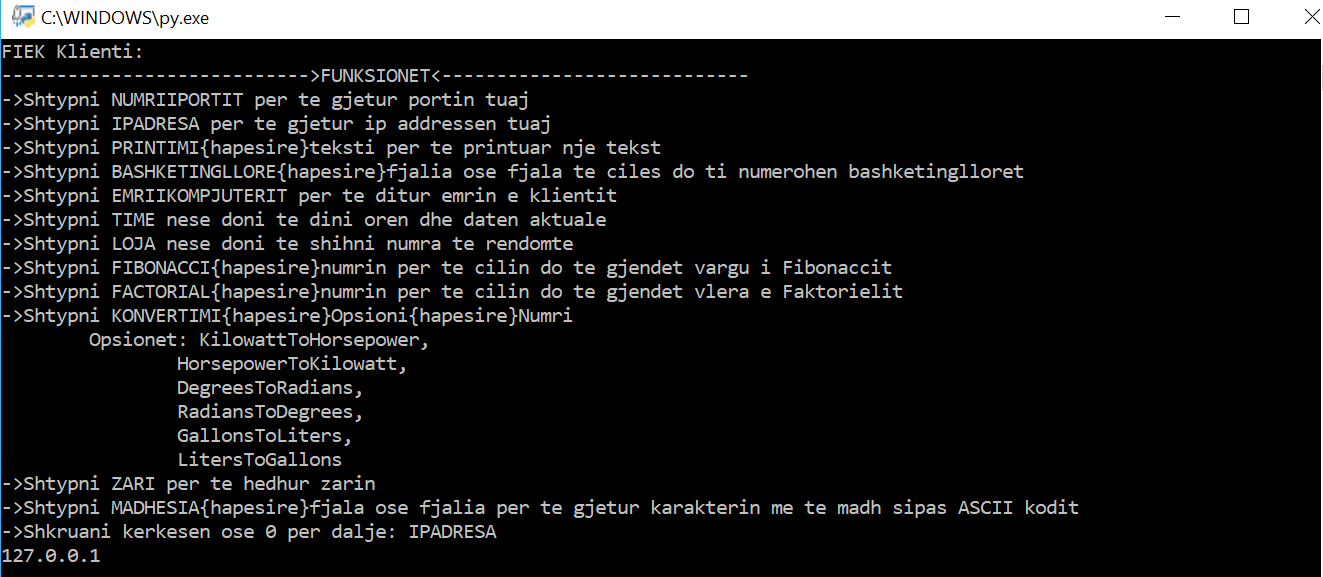
**Testimet me UDP-Protokol**

**Lidhjet**

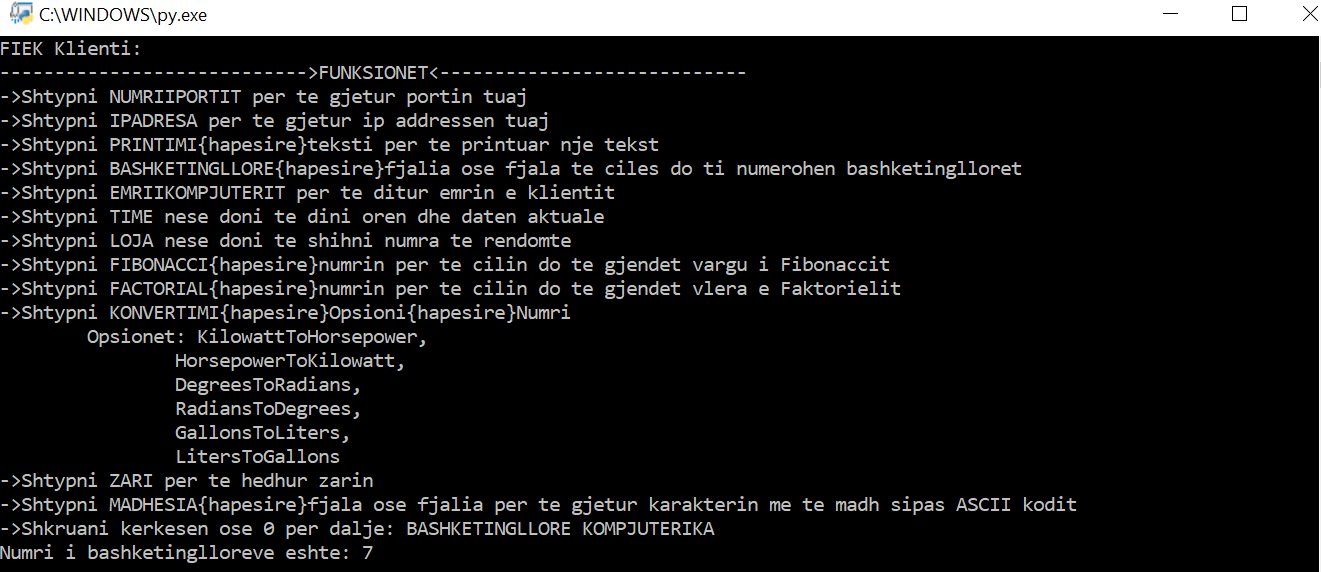
****

**NUMRIIPORTIT**

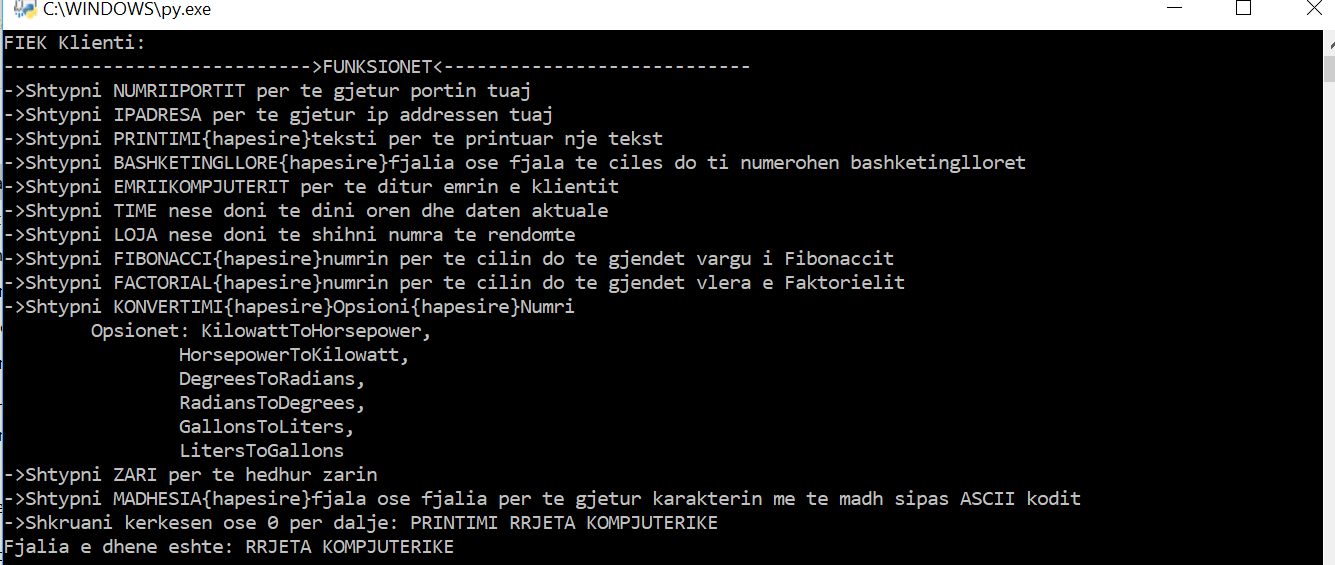
****

**IPADRESA  
**

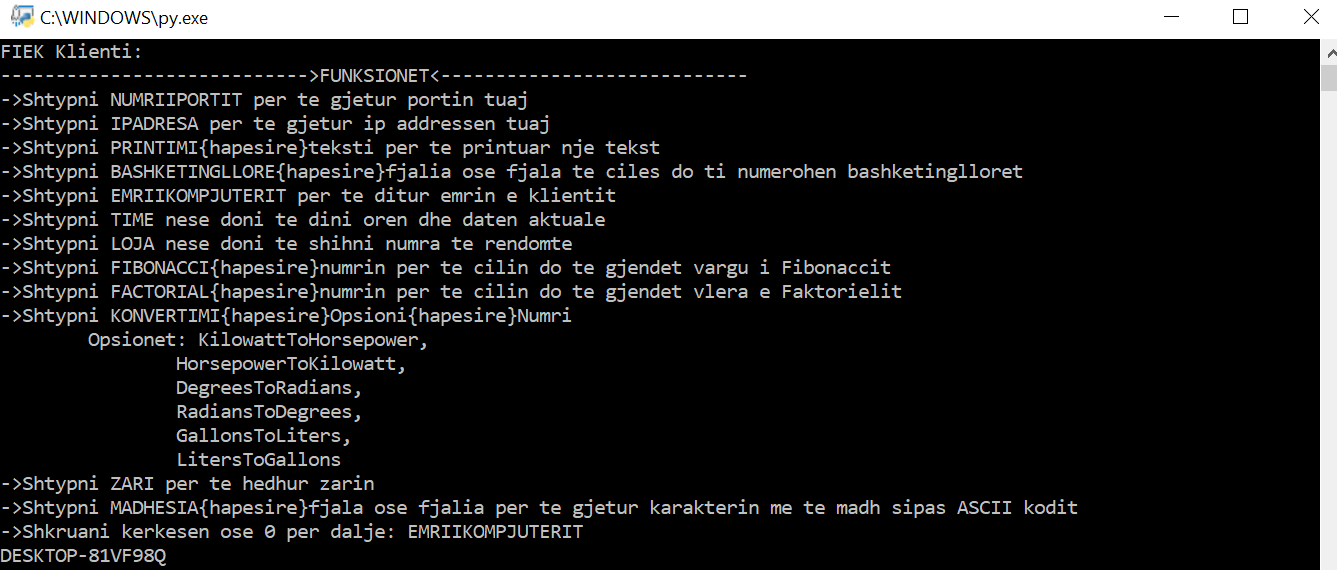
**BASHKETINGLLORE**

****

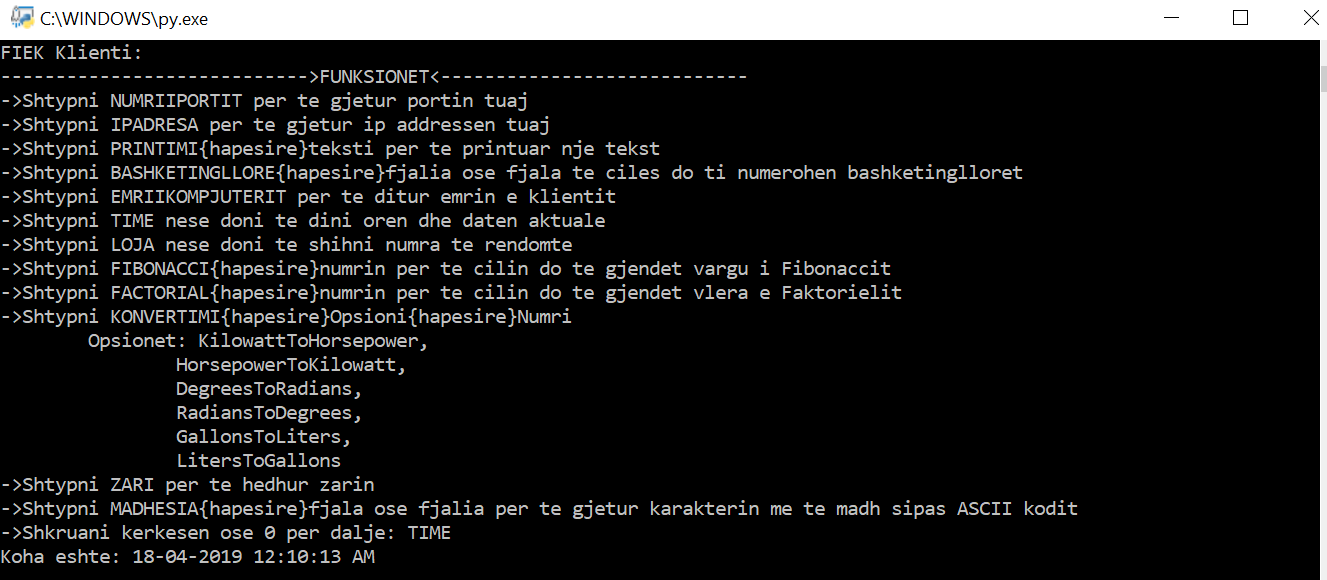
**PRINTIMI**

****

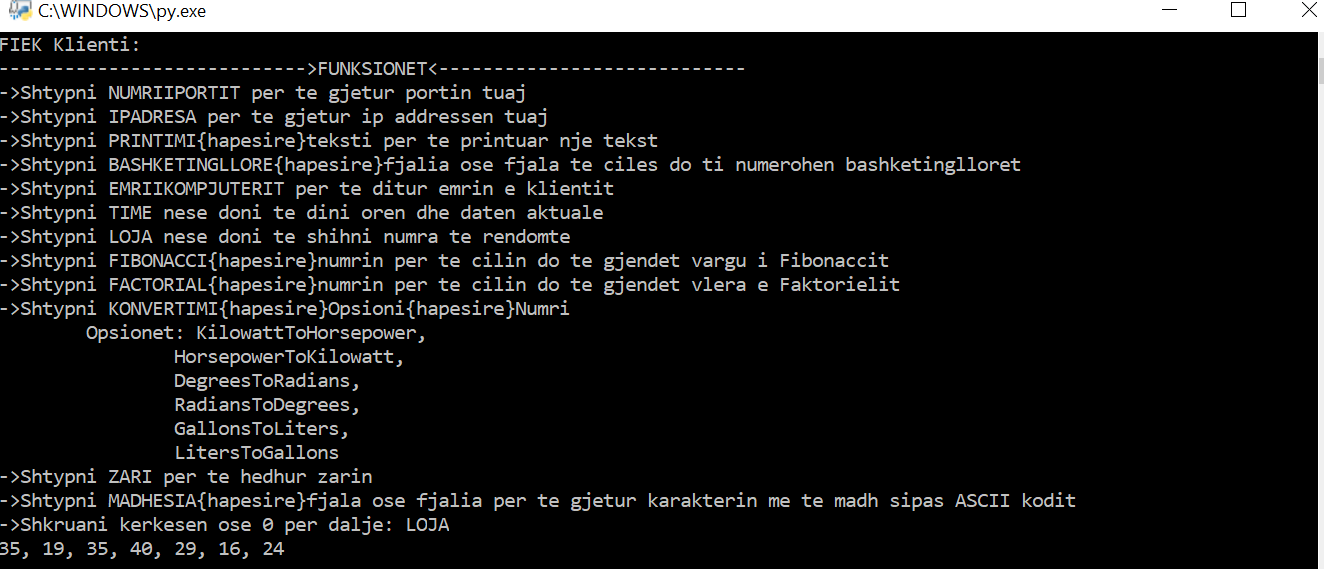
**EMRIIKOMPJUTERIT**

****

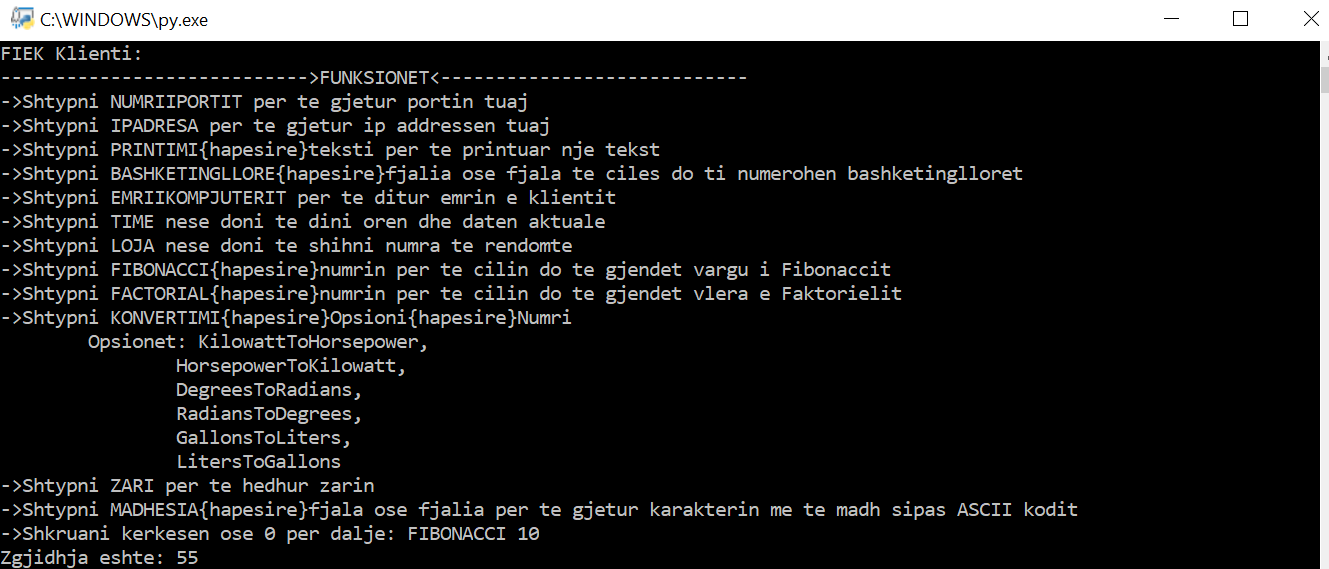
**TIME**

****

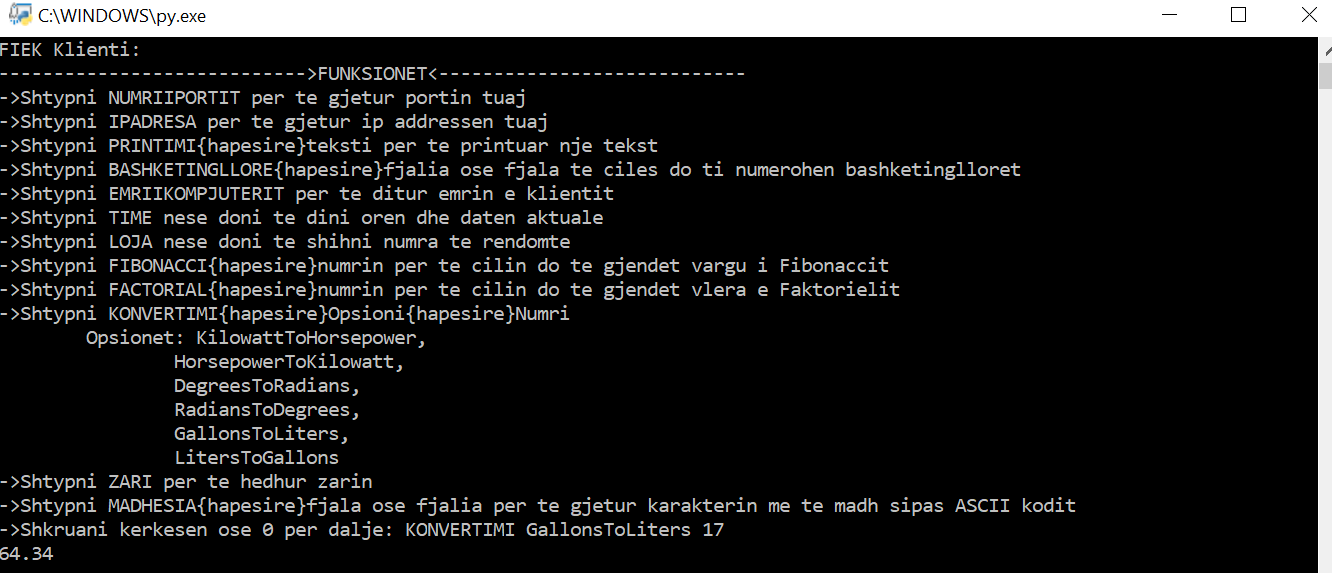
**LOJA**

****

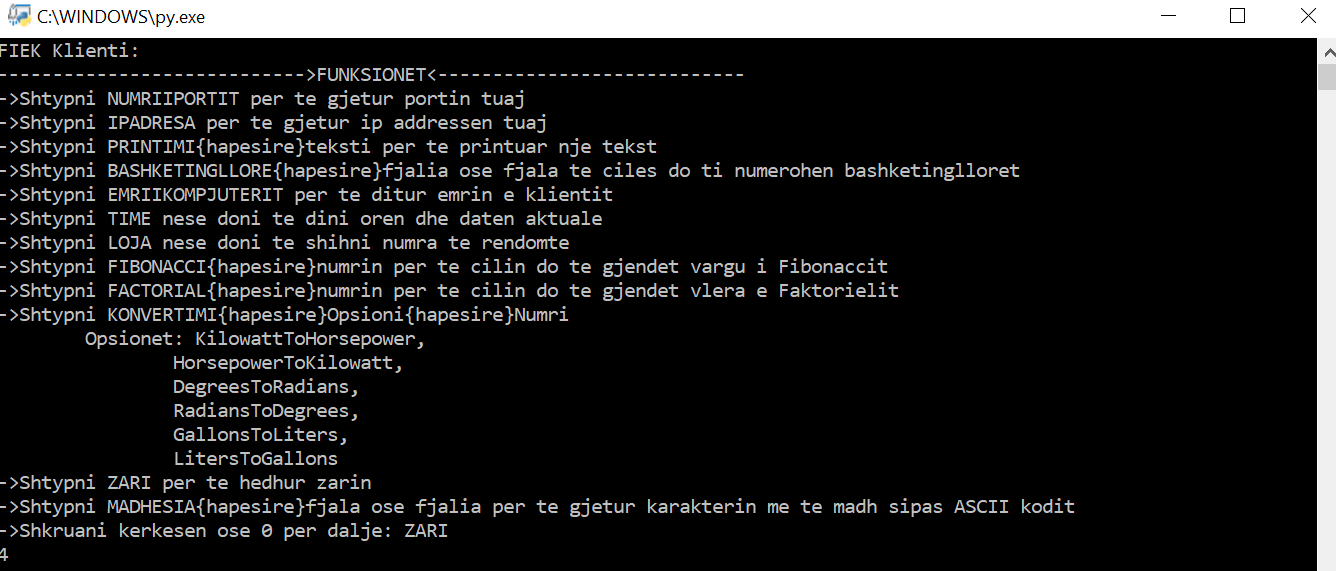
**FIBONACCI**

****

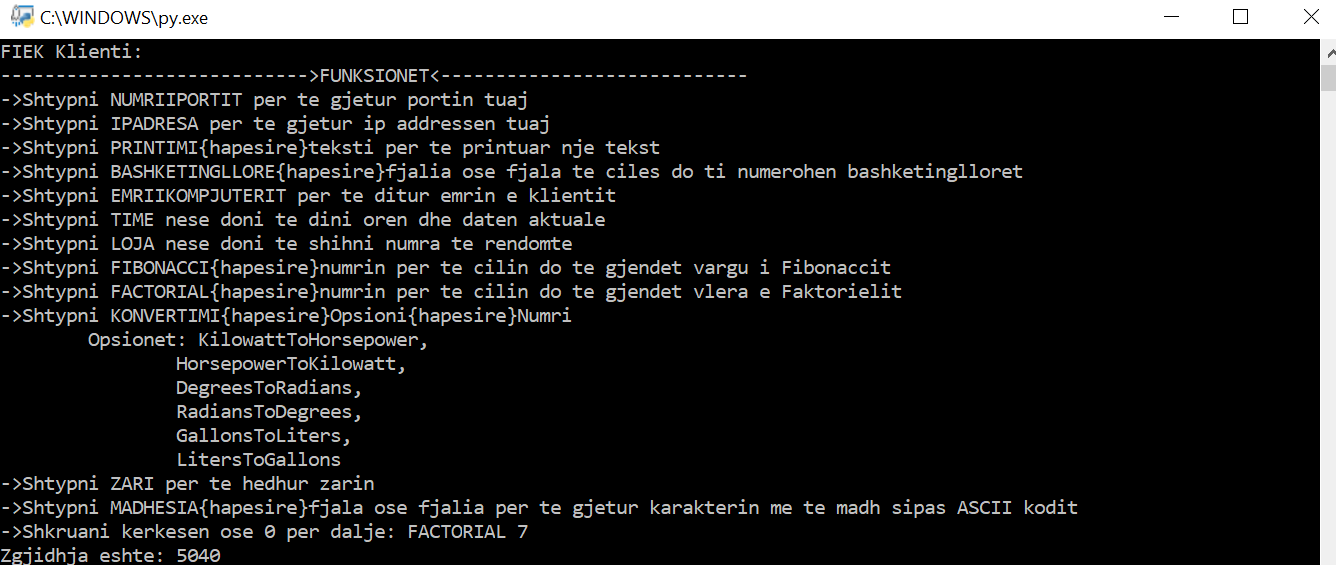
**KONVERTIMI**

****

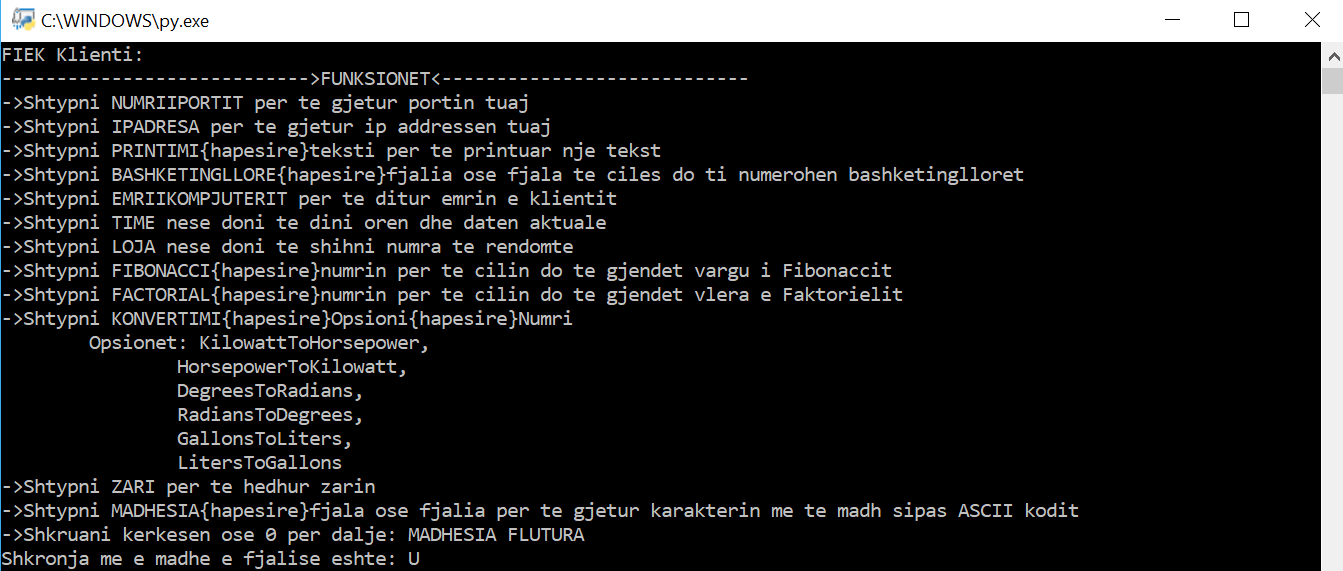
**ZARI**

****

**FACTORIAL**

****

**MADHESIA**

****

# Kodi

## Source Code per ServerSocketTCP

# Importimi i librarise socket per krijimin e socketave per lidhje

from socket import \*

# Importimi i librarise per perdorimin e funksionit start\_new\_thread

from \_thread import \*

import datetime

import random

import math

# Porti i serverit, ne te cilin do te pritet per komunikim

serverPort=12000

# Krijohet socketi ku edhe tregohet nga AF\_INET dhe SOCK\_STREAM se do te perdorim IPv4

serverSocket=socket(AF\_INET,SOCK\_STREAM)

# Startohet serveri, funksioni bind merr nje argument liste, elementi i pare eshte ip, i dyti eshte porti ku do te pret serveri per lidhje

serverSocket.bind(('', serverPort))

print('Serveri u startua ne localhost: '+str(serverPort))

print('-----------------------------------')

# Maksimumi i clientave qe mund te lidhen me server

serverSocket.listen(5)

print('Serveri eshte i gatshem te pranoje kerkese ') #

print('------------------------------------------')

#Funksionet e llogaritjeve

#Metoda Bashketingllore

def bashketingellore(fjalia):

bashketingellore = ['B', 'C','Ç' 'D','DH','F','G','GJ','H', 'J','K','L','LL','M','N',

'NJ','P','Q', 'R','RR','S','SH','T','TH','V','X','XH','Z','ZH',

'b','c','ç','d','dh','f','g','gj','h','j','k','l','ll','m','n',

'nj','p','q','r','rr','s','sh','t','th','v','x','xh','z','zh']

count=0

for shkronja in fjalia:

if(shkronja in bashketingellore):

count=count+1

return count

#Metoda EMRIIKOMPJUTERIT

def Host():

try:

host=gethostname()

return host

except socket.error:

return "Emri i hostit nuk mund te gjendet!"

#Metoda Koha

def timenow():

time=datetime.datetime.now()

return time.strftime("%d-%m-%Y %I:%M:%S %p")

#Metoda Loja

def loja():

listaNumrave=""

for number in range(0,7):

numriRandom=random.randint(1,49)

listaNumrave+=str(numriRandom)

if number!=6:

listaNumrave+=", "

return listaNumrave

#Metoda Fibonacci

def fibonacci(num):

if num==0:

return 0;

elif num == 1:

return 1;

elif num < 0:

return "Numri eshte me i vogel se zero!"

else:

return fibonacci(num-1)+fibonacci(num-2);

#Metoda Konvertimi

def konvertimi(opsioni, vlera):

if opsioni=="KilowattToHorsepower":

rezultati=vlera\*1.34

elif opsioni=="HorsepowerToKilowatt":

rezultati=vlera/1.34

elif opsioni=="DegreesToRadians":

rezultati = (vlera \* (math.pi/ 180))

elif opsioni=="RadiansToDegrees":

rezultati = (vlera \* (180/math.pi))

elif opsioni=="GallonsToLiters":

rezultati = vlera \* 3.785

elif opsioni=="LitersToGallons":

rezultati = vlera/3.785

else:

rezultati="Gabim"

return '%.2f' %(rezultati)

#Metoda Faktorieli

def factorial(nr):

if nr<=1:

return 1

else:

return nr\*factorial(nr-1)

#Metoda Madhesia

def madhesia(fjalia):

shkronja = 'A'

fjalia=fjalia.upper()

for S in fjalia:

if ord(S) > ord(shkronja):

shkronja = S

return shkronja

'''

Funksioni client\_handler ka 2 parametra, i pari eshte socketi i lidhjes me nje client i dyti esht adresa e klientit,

Ky funksion ekzekutohet ne thread tjeter per secilin klient

'''

def client\_handler(connectionSocket, addr):

# Loopi i cili pret per informacione nga klientet dhe kthen pergjigjen e duhur per kerkesen e dhene

while 1:

# Pranon te dhena nga klienti

fjalia = connectionSocket.recv(128)

if fjalia.decode('UTF-8').upper() == 'KOMANDAT':

komandat = ''

komandat += '---------------------------->FUNKSIONET<----------------------------\n'

komandat += '->Shtypni IPADRESA per te gjetur ip addressen tuaj\n'

komandat += '->Shtypni NUMRIIPORTIT per te gjetur portin tuaj\n'

komandat += '->Shtypni BASHKETINGLLORE per te numruar numrin e bashketinglloreve ne nje fjale ose fjali\n'

komandat += '->Shtypni PRINTIMI per te printuar ate qe keni shenuar\n'

komandat += '->Shtypni EMRIIKOMPJUTERIT per te ditur emrin e klientit\n'

komandat += '->Shtypni TIME nese doni te dini oren dhe daten aktuale\n'

komandat += '->Shtypni LOJA nese doni te shihni numra te rendomte\n'

komandat += '->Shtypni FIBONACCI pastaj numrin per te cilin gjendet vargu i Fibonaccit\n'

komandat += '->Shtypni KONVERTIMI per te bere konvertimin e numrave\n'

komandat += '->Shtypni ZARI per te hedhur zarin\n'

komandat += '->Shtypni FACTORIAL per te gjetur faktorielin e nje numri \n'

komandat += '->Shtypni MADHESIA per te gjetur karakterin me te madh sipas ASCII kodit\n'

connectionSocket.send(komandat.encode('UTF-8'))

fjalia = connectionSocket.recv(1024)

# Ruan ne nje variabel kerkesen me shkronja te medha

FjaliaMeShkronjaTeMedha=fjalia.upper()

# Kerkesa e dhene nga klienti ruhet ne nje variabel tjeter rcvCMD e dekoduar

rcvCMD = FjaliaMeShkronjaTeMedha.decode('UTF-8')

if rcvCMD == 'PRINTIMI':

# Pas pranimit te fjales PRINTO, pret per nje fjale tjeter nga klienti

mesazhi = connectionSocket.recv(128).decode('UTF-8')

mesazhi = 'Fjalia e dhene eshte: ' + mesazhi

connectionSocket.send(mesazhi.encode('UTF-8'))

elif rcvCMD == 'EMRIIKOMPJUTERIT':

response = 'Emri i klientit eshte: ' + Host()

connectionSocket.send(response.encode('UTF-8'))

elif rcvCMD == 'IPADRESA':

connectionSocket.send(str.encode(addr[0]))

elif rcvCMD == 'NUMRIIPORTIT':

response = 'Porti i serverit eshte ' + str(addr[1])

connectionSocket.send(str.encode(response))

elif rcvCMD == 'BASHKETINGLLORE':

fjalia = connectionSocket.recv(128).decode('UTF-8')

pergjigja = 'Numri i bashketingllore eshte: ' + str(bashketingellore(fjalia))

connectionSocket.send(pergjigja.encode('UTF-8'))

elif rcvCMD == 'TIME':

koha = 'Koha eshte: ' + str(timenow())

connectionSocket.send(koha.encode('UTF-8'))

elif rcvCMD == 'LOJA':

response = 'Numrat e shtypur nga intervali [1,49] jane: ( ' + loja() + ' )'

connectionSocket.send(response.encode('UTF-8'))

elif rcvCMD == 'ZARI':

response = 'Zari i hedhur ka vleren: ' + str(random.randint(1, 6))

connectionSocket.send(response.encode('UTF-8'))

elif rcvCMD == 'FIBONACCI':

nr = connectionSocket.recv(128).decode('UTF-8')

print('FIBONACCI')

print('Numri i dhene eshte: ' + nr)

num=int(nr)

connectionSocket.send(str(fibonacci(num)).encode('UTF-8'))

elif rcvCMD == 'FACTORIAL':

nr = connectionSocket.recv(128).decode('UTF-8')

print('FACTORIAL')

print('Numri i dhene eshte: ' + nr)

nr=int(nr)

connectionSocket.send(str(factorial(nr)).encode('UTF-8'))

elif rcvCMD == 'MADHESIA':

mesazhi = connectionSocket.recv(128).decode('UTF-8')

mesazhi = 'Shkronja me e madhe e fjalise eshte: ' + madhesia(mesazhi)

connectionSocket.send(mesazhi.encode('UTF-8'))

elif rcvCMD == 'KONVERTIMI':

opsioni = connectionSocket.recv(128).decode('UTF-8')

print('Zgjedhja e bere eshte: ' +opsioni)

num = connectionSocket.recv(128).decode('UTF-8')

numri=int(num)

connectionSocket.send(str(konvertimi(opsioni,numri)).encode('UTF-8'))

elif rcvCMD == '0':

break

else:

connectionSocket.send('Funksioni nuk ekziston'.encode('UTF-8'))

print('Klienti ' + str(addr) + ' mbylli lidhjen.')

connectionSocket.close()

while True:

# Pret lidhje nga nje klient dhe e ruan ate lidhje ne var connectionSocket me te cilen variabel mund te komunikojme me klientin

connectionSocket,addr=serverSocket.accept()

print('Klienti me ip ' + str(addr[0]) + ' dhe port ' + str(addr[1]) + ' krijoi nje lidhje')

# Krijon thread te ri per klientin i cili eshte pranuar nga serverSocket.accept(), dhe ekzekutohet funksioni client\_handler me parametrat e klientit perkates

start\_new\_thread(client\_handler, (connectionSocket, addr))

connectionSocket.close()

## Source Code per ClientSocketTCP

import socket

serverName="localhost"

serverPort=12000

# Krijohet nje socket te cilin e perdorim per lidhje me server

s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)

# Bejme kerkesen per lidhje ne serverin me emrin serverName dhe portin serverPort

s.connect((serverName,serverPort))

print('FIEK Klienti: ')

# Loopi i cili na pret per te shkruar kerkesen dhe ia dergon ate serverit pastaj pranon pergjigje dhe e priton ate

while True:

# Dergon kerkesen per komanda dhe i printon ato

s.sendall(str.encode('komandat'))

# 1024 per arsye se komandat jane me te medha se 128

komandat = s.recv(1024).decode('UTF-8')

print(komandat)

var=input('->Shkruani kerkesen ose numrin 0 per dalje: ')

mesazhi = var.upper()

s.sendall(str.encode(mesazhi))

if mesazhi == 'PRINTIMI' or mesazhi == 'BASHKETINGLLORE' or mesazhi == 'MADHESIA':

# Jepet nje fjali tjeter

fjalia = input('Shkruani nje fjali: ')

# E dergon fjaline te encoduar ne server

s.sendall(str.encode(fjalia))

# Pranon pergjigjen nga serveri dhe e dekodon

data=s.recv(128).decode('UTF-8')

# Printon pergjigjen e serverit

print(data)

elif mesazhi == 'FIBONACCI':

nr = input('Shkruani nje numer: ')

s.sendall(str.encode(nr))

data=s.recv(128).decode('UTF-8')

print('Fibonacci : '+data)

elif mesazhi=='FACTORIAL':

nr=input('Shkruani nje numer pozitiv:')

s.sendall(str.encode(nr))

data=s.recv(128).decode('UTF-8')

print('Factorial : '+data)

elif mesazhi == 'KONVERTIMI':

print('\t->KilowattToHorsepower')

print('\t->HorsepowerToKilowatt')

print('\t->DegreesToRadians')

print('\t->RadiansToDegrees')

print('\t->GallonsToLiters')

print('\t->LitersToGallons')

opsioni = input('Zgjedh konvertimin ne njeren nga format me siper: ')

s.sendall(str.encode(opsioni))

numri = input('Shkruaj numrin qe doni ta konvertoni: ')

s.sendall(str.encode(numri))

data=s.recv(128).decode('UTF-8')

print('Numri i konvertuar eshte: '+data)

elif mesazhi == '0':

break

else: # Nese nuk ka parametra te komanda vetem dergohet dhe kthehet pergjigja

data=s.recv(128).decode('UTF-8')

print(data)

print()

s.close()

## Source Code per ServerSocketUDP

from socket import \*

from \_thread import \*

import datetime

import random

import math

serverPort=12000

serverSocket=socket(AF\_INET,SOCK\_DGRAM)

serverSocket.bind(('', serverPort))

print('Serveri u startua ne localhost: '+str(serverPort))

print('-----------------------------------')

print('Serveri eshte i gatshem te pranoje kerkesa ')

print('------------------------------------------')

#Definimi i metodave

#Metoda Bashketingllore

def bashketingellore(fjalia):

bashketingellore = ['B', 'C','C' 'D','DH','F','G','GJ','H', 'J','K','L','LL','M','N',

'NJ','P','Q', 'R','RR','S','SH','T','TH','V','X','XH','Z','ZH',

'b','c','c','d','dh','f','g','gj','h','j','k','l','ll','m','n',

'nj','p','q','r','rr','s','sh','t','th','v','x','xh','z','zh']

count=0

for shkronja in fjalia:

if(shkronja in bashketingellore):

count=count+1

return count

#Metoda EMRIIKOMPJUTERIT

def Host():

try:

host=gethostname()

return host

except socket.error:

return "Emri i hostit nuk mund te gjendet!"

#Metoda Koha

def timenow():

time=datetime.datetime.now()

return time.strftime("%d-%m-%Y %I:%M:%S %p")

#Metoda Loja

def loja():

listaNumrave=""

for number in range(0,7):

numriRandom=random.randint(1,49)

listaNumrave+=str(numriRandom)

if number!=6:

listaNumrave+=", "

return listaNumrave

#Metoda Fibonacci

def fibonacci(num):

if num==0:

return 0;

elif num == 1:

return 1;

elif num < 0:

return "Numri eshte me i vogel se zero!"

else:

return fibonacci(num-1)+fibonacci(num-2);

#Metoda Konvertimi

def konvertimi(opsioni, vlera):

if opsioni=="KilowattToHorsepower":

rezultati=vlera\*1.34

elif opsioni=="HorsepowerToKilowatt":

rezultati=vlera/1.34

elif opsioni=="DegreesToRadians":

rezultati = (vlera \* (math.pi/ 180))

elif opsioni=="RadiansToDegrees":

rezultati = (vlera \* (180/math.pi))

elif opsioni=="GallonsToLiters":

rezultati = vlera \* 3.785

elif opsioni=="LitersToGallons":

rezultati = vlera/3.785

else:

rezultati="Gabim"

return '%.2f' %(rezultati)

#Metoda Faktorieli

def factorial(nr):

if nr<=1:

return 1

else:

return nr\*factorial(nr-1)

#Metoda Madhesia

def madhesia(fjalia):

shkronja = 'A'

fjalia=fjalia.upper()

for S in fjalia:

if ord(S) > ord(shkronja):

shkronja = S

return shkronja

def client\_handler(data, addr):

fjalia = data

# Ruan ne nje variabel kerkesen me shkronja te medha

FjaliaMeShkronjaTeMedha=fjalia.upper()

# Kerkesa e dhene nga klienti ruhet ne nje variabel tjeter rcvCMD e dekoduar

rcvCMD = FjaliaMeShkronjaTeMedha.decode('UTF-8')

if 'PRINTIMI' in rcvCMD:

fjalia = fjalia.decode('UTF-8')

n\_arr = fjalia.split(' ')

mesazhi = ''

for i in range(1, len(n\_arr)):

mesazhi += n\_arr[i] + ' '

mesazhi = 'Fjalia e dhene eshte: ' + mesazhi

serverSocket.sendto(mesazhi.encode('UTF-8'), addr)

elif rcvCMD == 'EMRIIKOMPJUTERIT':

serverSocket.sendto(Host().encode('UTF-8'), addr)

elif rcvCMD == 'IPADRESA':

serverSocket.sendto(str(addr[0]).encode('UTF-8'), addr)

elif rcvCMD == 'NUMRIIPORTIT':

serverSocket.sendto(str(addr[1]).encode('UTF-8'), addr)

elif 'BASHKETINGLLORE' in rcvCMD:

fjalia = fjalia.decode('UTF-8')

n\_arr = fjalia.split(' ')

mesazhi = ''

for i in range(1, len(n\_arr)):

mesazhi += n\_arr[i] + ' '

pergjigja = 'Numri i bashketinglloreve eshte: ' + str(bashketingellore(mesazhi))

serverSocket.sendto(pergjigja.encode('UTF-8'), addr)

elif rcvCMD == 'TIME':

koha = 'Koha eshte: ' + str(timenow())

serverSocket.sendto(koha.encode('UTF-8'), addr)

elif rcvCMD == 'LOJA':

serverSocket.sendto(loja().encode('UTF-8'), addr)

elif rcvCMD == 'ZARI':

serverSocket.sendto(str(random.randint(1, 6)).encode('UTF-8'), addr)

elif 'FIBONACCI' in rcvCMD:

nr = rcvCMD.split(' ')[1]

num=int(nr)

serverSocket.sendto(str('Zgjidhja eshte: ' + str(fibonacci(num))).encode('UTF-8'), addr)

elif 'MADHESIA' in rcvCMD:

fjalia = fjalia.decode('UTF-8')

n\_arr = fjalia.split(' ')

mesazhi = ''

for i in range(1, len(n\_arr)):

mesazhi += n\_arr[i] + ' '

mesazhi = 'Shkronja me e madhe e fjalise eshte: ' + madhesia(mesazhi)

serverSocket.sendto(mesazhi.encode('UTF-8'), addr)

elif 'FACTORIAL' in rcvCMD:

nr = rcvCMD.split(' ')[1]

nr=int(nr)

serverSocket.sendto(str('Zgjidhja eshte: ' + str(factorial(nr))).encode('UTF-8'), addr)

elif 'KONVERTIMI' in rcvCMD:

command = fjalia.decode('UTF-8')

opsioni = command.split(' ')[1]

print('Zgjedhja e bere eshte: ' + opsioni)

num = command.split(' ')[2]

numri=int(num)

serverSocket.sendto(str(konvertimi(opsioni,numri)).encode('UTF-8'), addr)

elif rcvCMD == '0':

pass

else:

serverSocket.sendto('Funksioni nuk ekziston'.encode('UTF-8'), addr)

while True:

data,addr = serverSocket.recvfrom(128)

print('Adresa ' + str(addr) + ' dergoi nje informacion')

start\_new\_thread(client\_handler, (data, addr))

## Source Code per ClientSocketUDP

import socket

serverName="localhost"

serverPort=12000

s=socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM)

print('FIEK Klienti: ')

while True:

print('---------------------------->FUNKSIONET<----------------------------')

print('->Shtypni NUMRIIPORTIT per te gjetur portin tuaj')

print('->Shtypni IPADRESA per te gjetur ip addressen tuaj')

print('->Shtypni PRINTIMI{hapesire}teksti per te printuar nje tekst')

print('->Shtypni BASHKETINGLLORE{hapesire}fjalia ose fjala te ciles do ti numerohen bashketinglloret')

print('->Shtypni EMRIIKOMPJUTERIT per te ditur emrin e klientit')

print('->Shtypni TIME nese doni te dini oren dhe daten aktuale')

print('->Shtypni LOJA nese doni te shihni numra te rendomte')

print('->Shtypni FIBONACCI{hapesire}numrin per te cilin do te gjendet vargu i Fibonaccit')

print('->Shtypni FACTORIAL{hapesire}numrin per te cilin do te gjendet vlera e Faktorielit')

print('->Shtypni KONVERTIMI{hapesire}Opsioni{hapesire}Numri')

print('\tOpsionet: KilowattToHorsepower, \n\t\tHorsepowerToKilowatt, \n\t\tDegreesToRadians, \n\t\tRadiansToDegrees, \n\t\tGallonsToLiters, \n\t\tLitersToGallons')

print('->Shtypni ZARI per te hedhur zarin')

print('->Shtypni MADHESIA{hapesire}fjala ose fjalia per te gjetur karakterin me te madh sipas ASCII kodit')

var=input('->Shkruani kerkesen ose 0 per dalje: ')

mesazhi = var

s.sendto(str.encode(mesazhi), (serverName, serverPort))

if mesazhi != '0':

data=s.recv(128).decode('UTF-8')

print(data)

else:

break

print()

s.close()

# REFERENCAT

“Computer Networking – A top down approach” 7Ed.,2017 **James F. Kurose, Keith W. Ross**

<https://docs.python.org/3/>

<https://www.geeksforgeeks.org/socket-programming-python/>

<https://stackoverflow.com/>  
<https://www.python-course.eu/threads.php>